US009350702B2

(12) **United States Patent**
Faulk, Jr.

(10) **Patent No.: US 9,350,702 B2**
(45) **Date of Patent: May 24, 2016**

(54) **VIRTUAL INSERTION INTO A NETWORK**

(75) Inventor: **Robert L. Faulk, Jr.**, Roseville, CA (US)

(73) Assignee: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP**, Houston, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1130 days.

(21) Appl. No.: **12/707,046**

(22) Filed: **Feb. 17, 2010**

(65) **Prior Publication Data**

US 2011/0202675 A1 Aug. 18, 2011

(51) **Int. Cl.**
G06F 15/16 (2006.01)
H04L 29/06 (2006.01)
G06F 12/00 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 63/02*** (2013.01); ***H04L 29/06*** (2013.01)

(58) **Field of Classification Search**
USPC 709/220–240, 206, 250; 370/230–260; 705/112–138
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,036,229	B2 *	10/2011	Banerjee	370/397
8,892,708	B2 *	11/2014	Merrill et al.	709/223
2007/0005786	A1 *	1/2007	Kumar et al.	709/230
2007/0140128	A1 *	6/2007	Klinker et al.	370/238
2008/0046549	A1 *	2/2008	Saxena et al.	709/223
2009/0300605	A1 *	12/2009	Edwards et al.	718/1
2009/0303883	A1	12/2009	Kucharczyk et al.	

OTHER PUBLICATIONS

Anand Gorti and Vijoy Pandey, "OSF-Open Service Framework: An Integrated High-speed Load Balancing and Fow Steering Framework," In Proceedings of First Workshop on Data Center—Converged and Virtual Ethernet Switching, DC-Caves, 2009, pp. 1-6.

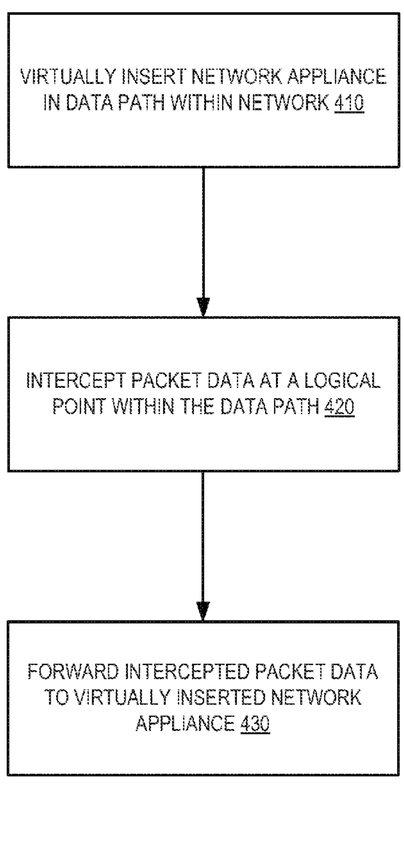
* cited by examiner

Primary Examiner — Zarni Maung
(74) *Attorney, Agent, or Firm* — Hewlett Packard Enterprise Patent Department

(57) **ABSTRACT**

A network appliance is virtually inserted in a data path within a network. Packet data that matches a criteria is intercepted at a logical point within the data path. The intercepted packet data is forwarded to an application running on the virtually inserted network appliance.

20 Claims, 5 Drawing Sheets



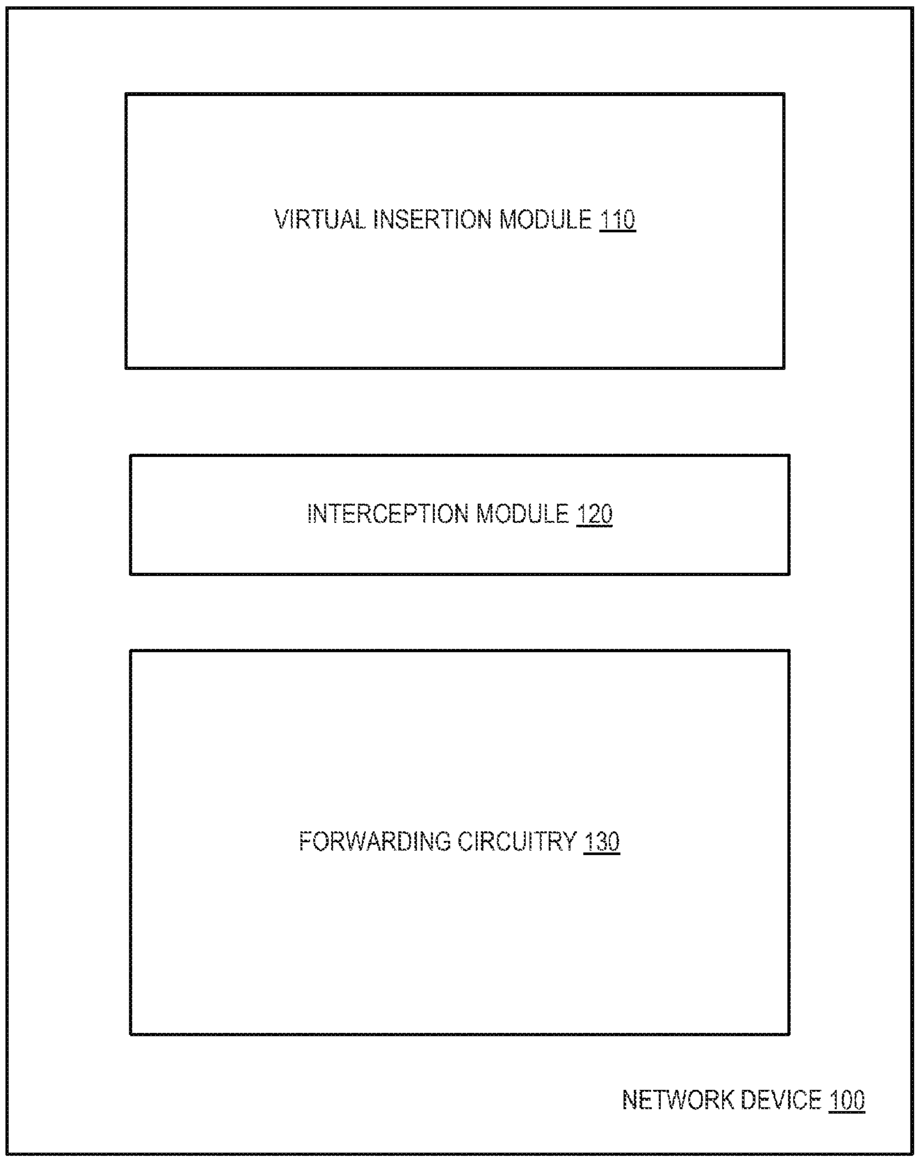

VIRTUAL INSERTION MODULE 110
INTERCEPTION MODULE 120
FORWARDING CIRCUITRY 130
NETWORK DEVICE 100

FIG. 1

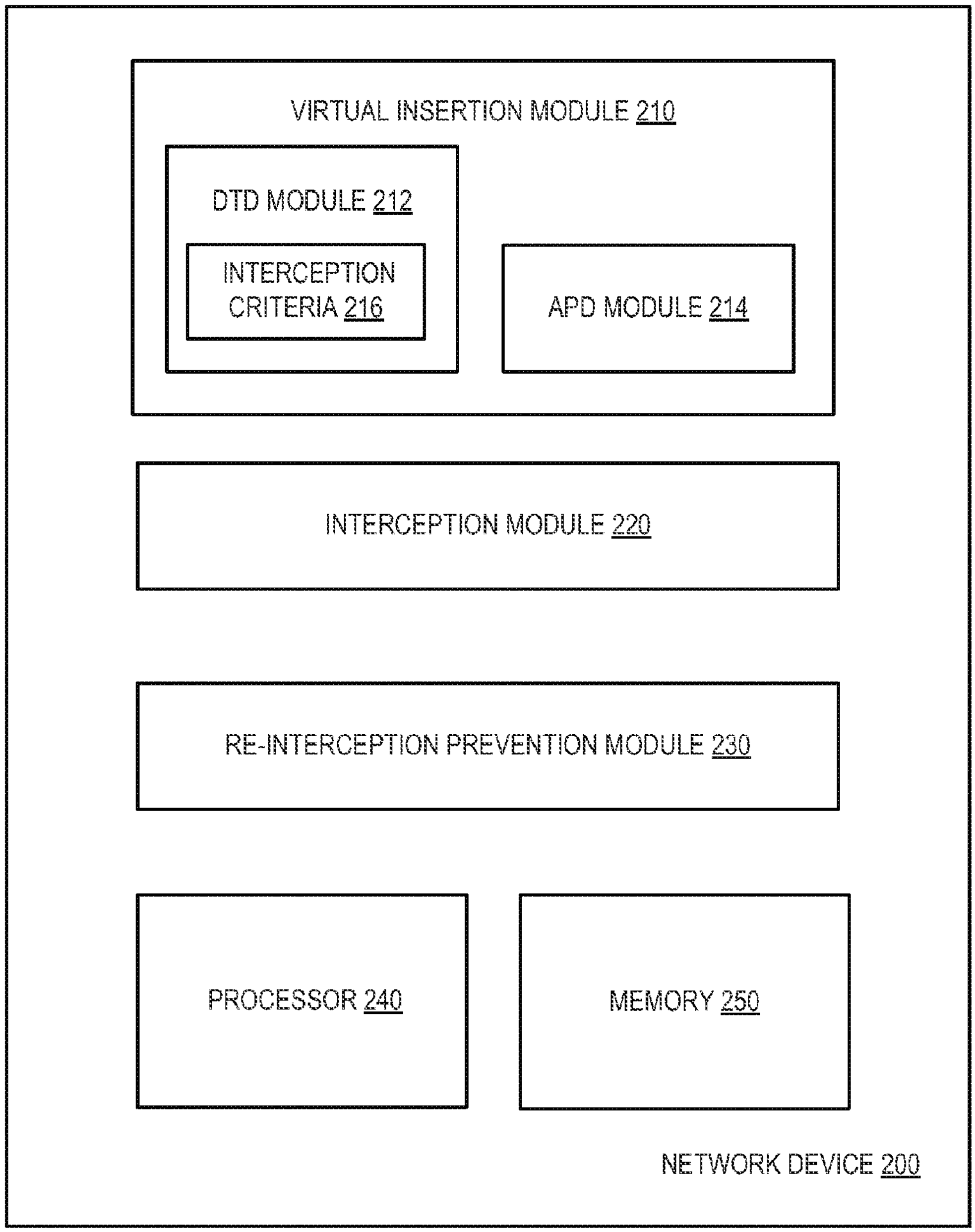
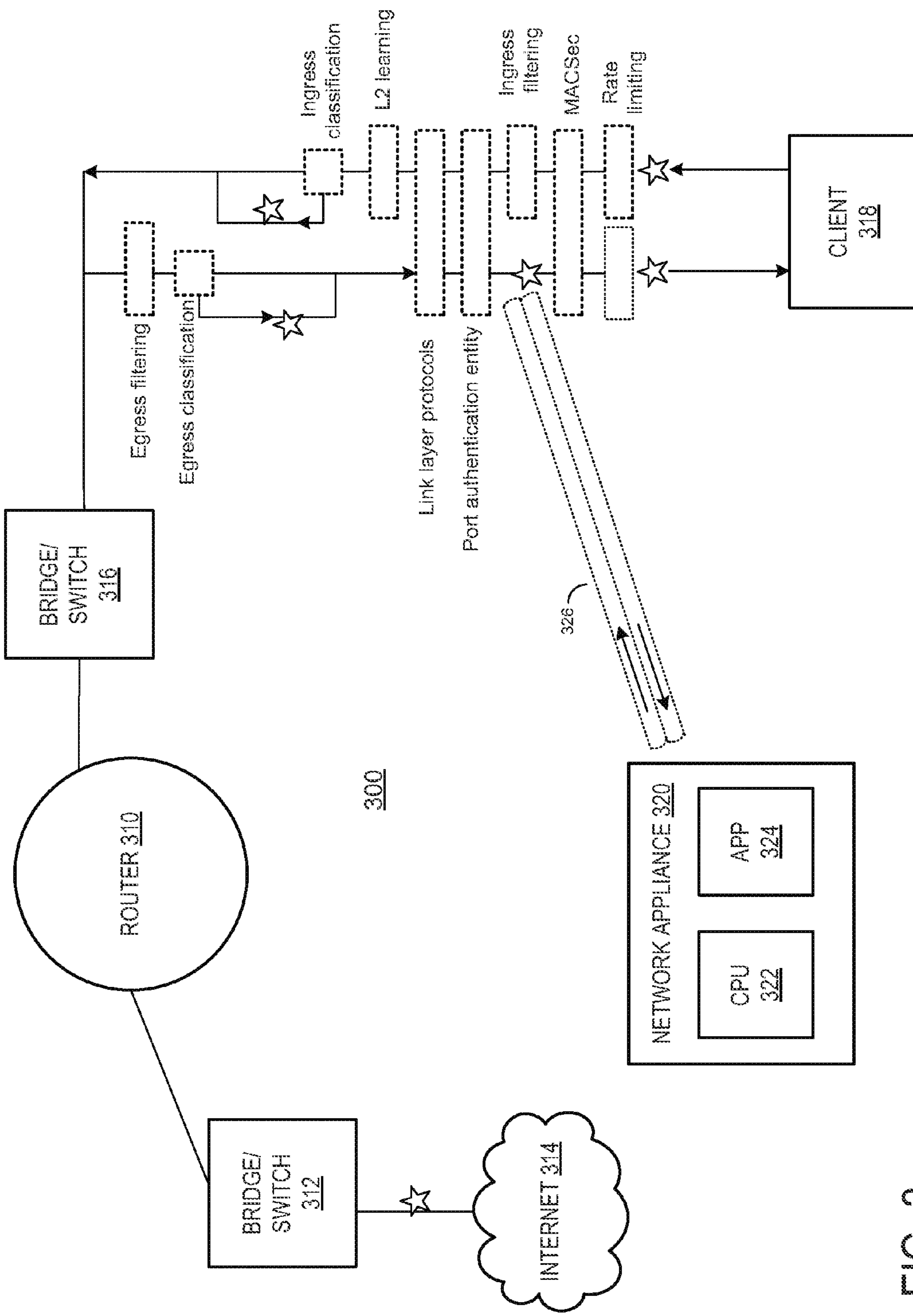

VIRTUAL INSERTION MODULE 210
DTD MODULE 212
INTERCEPTION CRITERIA 216
APD MODULE 214
INTERCEPTION MODULE 220
RE-INTERCEPTION PREVENTION MODULE 230
PROCESSOR 240
MEMORY 250
NETWORK DEVICE 200

FIG. 2



Ingress classification
L2 learning
Ingress filtering
MACSec
Rate limiting
Egress filtering
Egress classification
Link layer protocols
Port authentication entity
BRIDGE/ SWITCH 316
CLIENT 318
326
ROUTER 310
300
NETWORK APPLIANCE 320
APP 324
CPU 322
BRIDGE/ SWITCH 312
INTERNET 314

FIG. 3

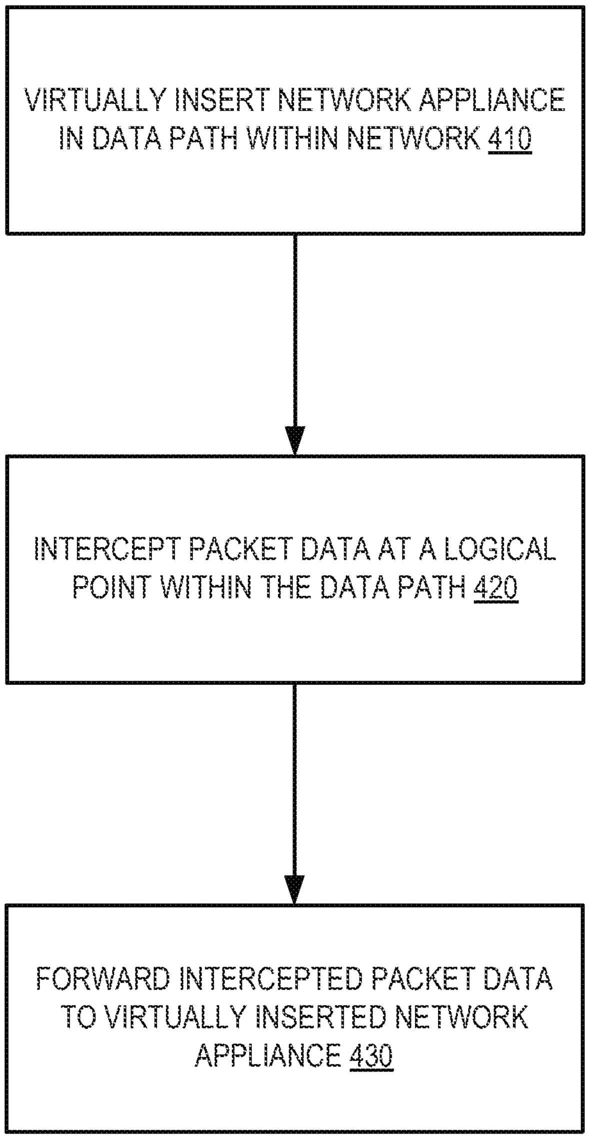

VIRTUALLY INSERT NETWORK APPLIANCE IN DATA PATH WITHIN NETWORK 410
INTERCEPT PACKET DATA AT A LOGICAL POINT WITHIN THE DATA PATH 420
FORWARD INTERCEPTED PACKET DATA TO VIRTUALLY INSERTED NETWORK APPLIANCE 430

FIG. 4

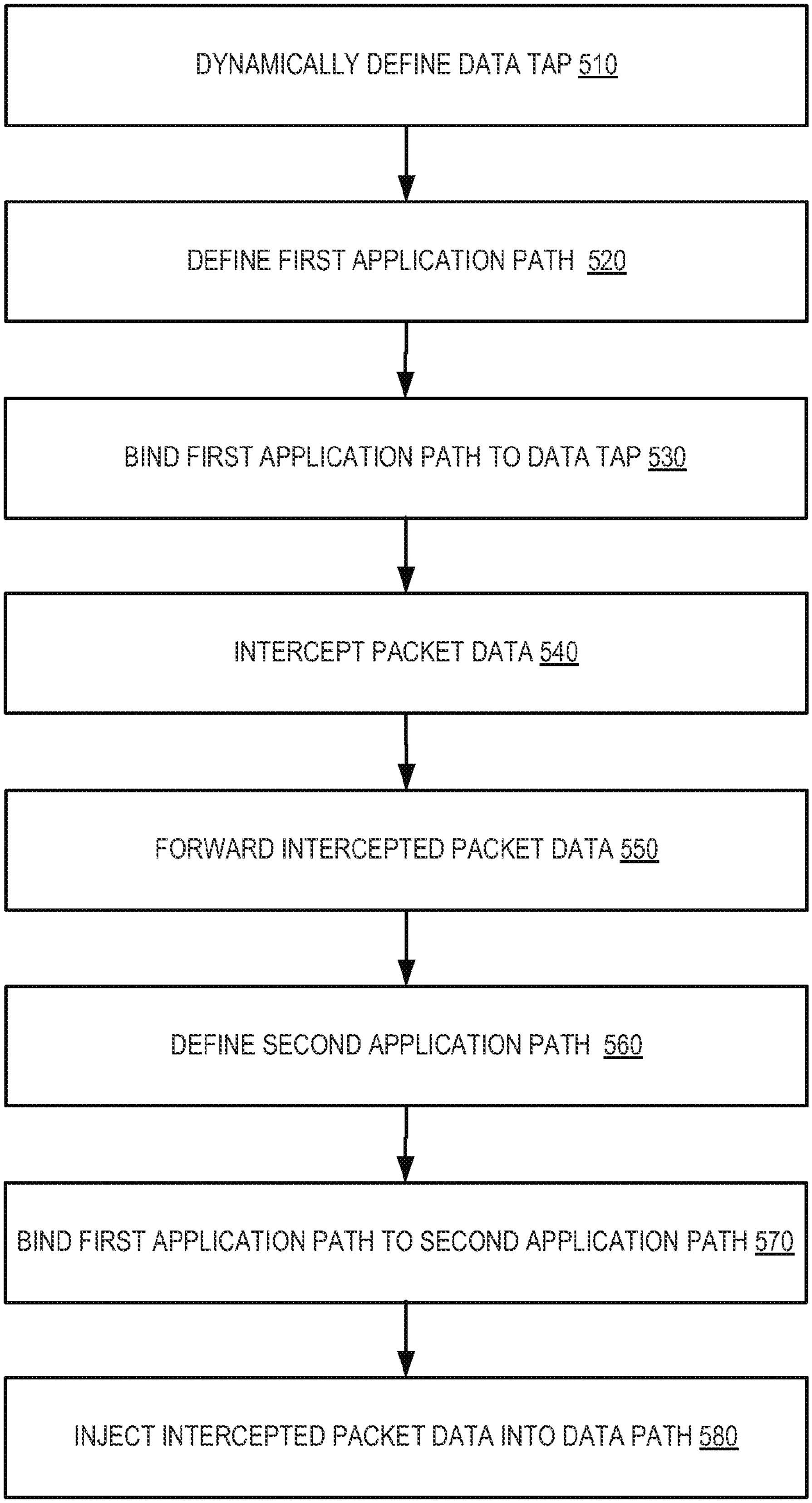

DYNAMICALLY DEFINE DATA TAP 510
DEFINE FIRST APPLICATION PATH 520
BIND FIRST APPLICATION PATH TO DATA TAP 530
INTERCEPT PACKET DATA 540
FORWARD INTERCEPTED PACKET DATA 550
DEFINE SECOND APPLICATION PATH 560
BIND FIRST APPLICATION PATH TO SECOND APPLICATION PATH 570
INJECT INTERCEPTED PACKET DATA INTO DATA PATH 580

FIG. 5

VIRTUAL INSERTION INTO A NETWORK

BACKGROUND

In a computer network, clients connect to network switches and/or routers, which frequently connect to the Internet. Various network appliances (e.g., Intrusion Prevention Systems (IPS), wide area network (WAN) accelerators, monitoring and/or troubleshooting devices, etc.) can be connected to the network for various purposes.

BRIEF DESCRIPTION OF DRAWINGS

The following description includes discussion of figures having illustrations given by way of example of implementations of embodiments of the invention.

FIG. **1** is a block diagram illustrating a device according to various embodiments.

FIG. **2** is a block diagram illustrating a device according to various embodiments.

FIG. **3** is a block diagram illustrating a system according to various embodiments.

FIG. **4** is a flow diagram of operation in a system according to various embodiments.

FIG. **5** is a flow diagram of operation in a system according to various embodiments.

DETAILED DESCRIPTION

Network appliances (e.g., IPS, WAN accelerators, monitoring and/or troubleshooting devices, etc.) can be added to a network by physically reconnecting network cables to get these appliances into the data path of the network. However, physically reconnecting network cables can be burdensome, especially in a dynamic network environment. In addition, physical connections may limit where in the data path an appliance may be inserted. Embodiments described herein enable virtual insertion of a network appliance into a network at desired point in the data path.

Policy based routing allows routed IP (Internet Protocol) packets matching a certain pattern to be forwarded to the designated next hop gateway (on a designated port and VLAN). Furthermore, with policy based routing, packets are modified in a specific manner—for example, the source MAC (Media Access Control) address is changed to be that of the router, the destination MAC address is changed to be that of the next hop gateway, and the VLAN is changed. Thus, in policy based routing, the next hop gateway is not given the original form of the packet. Various embodiments enable network appliances to receive the original unmodified form of a packet, or other forms. Furthermore, as described, various embodiments enable a network appliance to reinject a packet back into the forwarding data path (or simply data path) in its original unmodified form.

FIG. **1** is a block diagram illustrating a system according to various embodiments. Network device **100** may be any device that connects network segments and/or connects other devices to each other (e.g., an OSI layer 2 bridge, OSI layer 3 router, etc.). As illustrated, network device **100** includes a virtual insertion module **110**, an interception module **120**, and forwarding circuitry **130**. The components (i.e., modules, circuitry, etc.) shown in FIG. **1** may be logically and/or physical combined in various embodiments. In alternate embodiments, network device **100** may have more components, fewer components, and/or different components. The various components shown in FIG. **1** can be implemented as one or more software modules, hardware modules, special-purpose hardware (e.g., application specific hardware, application specific integrated circuits (ASICs), embedded controllers, hardwired circuitry, etc.), or some combination of these.

Virtual insertion module **110** controls virtual insertion of a network appliance into a forwarding data path. Specifically, virtual insertion module **110** inserts a network appliance into the data path based on a data tap. Data taps are described in more detail below. As used herein, a network appliance includes network devices that receive data (e.g., packet data), optionally perform some modification on the data (e.g., adding/changing a packet header, etc.), and optionally return the data back into the network. Examples of network appliances might include an Intrusion Prevention System (IPS), WAN (wide area network) accelerators, monitoring devices, troubleshooting devices, and the like. Networking devices that perform routing and/or switching functionality, along with the network appliance functionality described above, may also be considered network appliances in certain embodiments.

In various embodiments, virtual insertion module **110** defines data taps, defines application paths, and binds application paths to data taps, described in more detail below.

Interception module **120** intercepts data packets. Interception criteria may be based on, but are not limited to, raw ports, address-based forwarding, flow-based forwarding, ingress and/or egress classification, logical and/or physical ports, packet contents, packet flags, flow state, etc. In addition, a software agent running on network device **100** might be used as interception criteria. For example, the software agent might have its own criteria for receiving packets. Interception module **120** can be configured to intercept packets picked up by the software agent. In certain embodiments, intercepting packets via comparing packets against the criteria can be performed by hardware (e.g., on the network device ASIC) with no software involvement.

Forwarding circuitry **130** handles the forwarding of intercepted packet data to the network appliance.

FIG. **2** is a block diagram illustrating another system according to various embodiments. Similar to network device **100**, network device **200** includes a virtual insertion module **210** and an interception module **220**. Also included are a re-interception prevention module **230**, a processor **240** and memory **250**. The various components, modules, etc. shown in FIG. **2** can be implemented as one or more software modules, hardware modules, special-purpose hardware (e.g., application specific hardware, application specific integrated circuits (ASICs), embedded controllers, hardwired circuitry, etc.), or some combination of these.

Virtual insertion module **210** virtually inserts one or more network appliances into the forwarding data path using a data tap descriptor (DTD) module **212** and an application path descriptor (APD) module **214**.

DTD module **212** defines data taps. As used herein, a data tap describes a logical point in a forwarding data path for intercepting data packets. Given that many packet processing tasks are handled concurrently (e.g., by an ASIC) in a networking device, the logical point described by the data tap can be considered a function of priority and/or precedence. Using packet ingress as an example, a network device might give precedence to rate limiting over MAC (Media Access Control) security, or give precedence to port authentication over OSI Layer 2 (L2) learning. By defining a logical point within a group of prioritized packet processing operations, data taps allow a network appliance to virtually tap into any logical location in the data path. Rather than being limited to a fixed or default point in the data path, DTD module **212** can be dynamically updated to tap into multiple different logical locations in the data path.

In various embodiments, DTDs (data tap descriptors) are described using the GPPC (General Purpose Packet Control) MIB (management information base) and a policy-based forwarding CLI (command line interface) syntax. Other suitable schemes, protocols, syntaxes, etc. could be used in different embodiments. The GPPC MIB is one example of a tool that can specify where in the forwarding data path to locate a data tap. The data path may include a variety of logical locations—both ingress and egress—including, but not limited to, filtering, classification, L2 learning, link layer protocols, port authentication, MAC security, rate limiting, raw ports, software agents, NPU (network processing unit), etc. In one example, DTD module **212** may define an ingress data tap to intercept a pre-routed form (e.g., with respect to network device **200**) of a data packet and/or an egress data tap to intercept a post-routed form (e.g., with respect to network device **200**) of a data packet.

APD module **214** defines the application path between the network appliance and the data tap. More particularly, in various embodiments, APD module **214** defines the path between an application running on the network appliance and the data tap. In some embodiments, an application path may be defined for an application running on a device other than a network appliance. In various embodiments, application paths may be represented as network interfaces to x86 applications and/or switch agents. One or more tunnel drivers can be used to convert various encapsulations and/or unencapsulated data to network interfaces. Different interface configuration parameters may be used depending on the interface type (e.g., raw Ethernet, GRE/IPSec, PCI Express, proprietary encapsulations, etc.).

APDs (application path descriptors) may be configured using standard interface MIBs when standard encapsulations are used and proprietary MIBs for proprietary encapsulations. In various embodiments, APDs are bound to DTDs using the GPPC MIB and/or policy-based forwarding CLI syntax. Other schemes, protocols, syntaxes, etc. could be used for binding in different embodiments.

Interception module **220** intercepts data packets based on interception criteria **216** which can be dynamically updated. Interception criteria may be based on, but are not limited to, raw ports, address-based forwarding, flow-based forwarding, ingress and/or egress classification, logical and/or physical ports, packet contents, packet flags, flow state, etc. In addition, a software agent running on network device **200** might be used as an interception criterion. For example, the software agent might have its own criteria for receiving packets. Interception module **220** can be configured to intercept packets picked up by the software agent. In certain embodiments, intercepting packets via comparing packets against the criteria can be performed by hardware (e.g., on the device ASIC) with no software involvement.

In certain embodiments, interception module **220** prevents intercepted packets (or a portion of an intercepted packet) from being copied or sent to any location other than the destination defined by the corresponding APD. Re-interception prevention module **230** prevents previously intercepted packets from being re-intercepted by a data tap.

The various components, modules, functions, etc. described with respect to FIG. **2** may be implemented as instructions stored on a computer-readable storage medium (e.g., memory **250**) and executed by a processor (e.g., processor **240**).

FIG. **3** is a block diagram illustrating a network system according to various embodiments. As shown, network **300** includes a router **310**, multiple bridges **312**, **316**. Switch **312** connects to the Internet **314**. Client **318** connects to switch **316**. Starred locations in FIG. **3** indicate examples of logical locations for the data taps described herein. Other suitable logical locations (e.g., more, fewer, different locations) could be used in different embodiments. Network appliance **320** and/or application **324** are virtually inserted into the data path of network **300** via application path **326**. As shown, application path **326** is intended to illustrate an example of a path between network appliance **320** and a data tap (illustrated by a star). In various embodiments, application path **326** is a bi-directional path, however it could be a unidirectional path if network appliance **320** were used for monitoring only.

Various logical packet processing operations are shown in the path between switch **316** and client **318** for ease of illustration. In practice, such packet processing operations are actually performed within switch **316** in various embodiments. As discussed above, packet processing operations may be handled concurrently by a network device (e.g., switch **316**) and thus, the logical flow illustrated in FIG. **3** is based on a priority or precedence of operations—both for ingress and egress—with respect to switch **316**.

Network **300** may be implemented as separate network devices in some embodiments or some or all of network **300** could be implemented in a single ASIC or CPU in other embodiments.

FIG. **4** is a flow diagram of operation in a system according to various embodiments. An entity virtually inserts **410** a network appliance in a data path within a network. Inasmuch as the logical flow of the data exists within a network device, the network device (e.g., a virtual insertion module within the network device) acts as the inserting entity in various embodiments. As described above, the inserting entity can be implemented as one or more software modules, hardware modules, special-purpose hardware (e.g., application specific hardware, application specific integrated circuits (ASICs), embedded controllers, hardwired circuitry, etc.), or some combination of these.

By virtually inserting a network appliance, physical re-cabling and re-connecting can be avoided. In various embodiments, the virtual insertion is dynamic, meaning that the virtual location of the network appliance in the data path can be changed and updated (e.g., via data tap descriptors and application path descriptors) without the need to physically move the network appliance. The virtual insertion is also dynamic in that the data tap location may be changed and updated.

The network device intercepts **420** packet data at a logical point within the data path on the network based, at least in part, on a criterion. The criterion (or criteria) could be flow-based, port-based, classification-based, or based on any other suitable packet-related attribute.

In various embodiments, the network device forwards **430** intercepted packet data to an application running on the virtually inserted network appliance. In alternate embodiments, intercepted packet data may be forwarded to any location capable of processing (e.g., with a processing unit) the packet data.

FIG. **5** is a flow diagram of operation in a system according to various embodiments. To effectuate the virtual insertion described above, the system (or component, module, etc. thereof) dynamically defines **510** a data tap that describes a logical point within a data path to intercept data. For example, a data tap might specify a logical data path point between egress filtering and egress classification. Or, in another example, a data tap might specify the logical point in the data path between port rate limiting and the application of MAC security. Other logical points in the data path could be described by a data tap. Data taps are dynamic in various embodiments given that they may be updated and/or changed, for example, based on network conditions or other suitable factors.

The system defines **520** at least a first application path from a data tap to an application running on a virtually inserted network appliance. Application paths can be defined as network interfaces using MIBs, tunnel drivers, encapsulations, DMAs (direct memory accesses), or other suitable techniques.

The system binds **530** the first application path to the data tap. The binding combination of the application path and the data tap results in the virtual insertion of the network appliance at the logical location defined by the data tap.

Having virtually inserted the network appliance via binding an application path to a data tap, the system intercepts **540** packet data at the logical point in the data path defined by the data tap. Intercepted data is forwarded **550** to the virtually inserted network appliance. In various embodiments, the intercepted data is forwarded to an application running on the network appliance. In other embodiments, the intercepted data may be forwarded elsewhere on the network.

In certain embodiments, the system may define **560** a second application path from the data tap. Binding **570** the first application path to the second application path allows different application to be chained together at a particular data tap.

Intercepted packet data may be processed, modified, etc. by the application (or network appliance, etc.) receiving the intercepted data. In various embodiments, intercepted packets are injected **580** back into the data path (e.g., by the virtually inserted network appliance). In some embodiments, packets are injected in a pre-interception format. In other words, packets are injected such that there is no indication that they were intercepted in the first place. In other embodiments, packets are injected back into the data path in some other format.

The invention claimed is:

1. A method, comprising:

virtually inserting, by a processor, a network appliance in any one of a plurality of logical points within a data path of a network, including defining a data tap that describes one of the plurality of logical points within the data path to intercept data and defining a first application path between the data tap and the network appliance, wherein the network appliance is virtually inserted into a location in the data path of the network without physically connecting the network appliance into the location in the data path;

intercepting, by the processor, packet data at the described logical point within the data path of the network when the packet data matches a criteria; and

forwarding, by forwarding circuitry, the intercepted packet data to a first application running on the network appliance.

2. The method of claim **1**, wherein virtually inserting the network appliance in the data path further comprises:

binding the first application path to the data tap.

3. The method of claim **2**, further comprising:

defining a second application path between a second application running on the network appliance and the data tap; and

binding the first application path to the second application path.

4. The method of claim **1**, further comprising:

returning the intercepted packet data from the network appliance to the data path in a pre-intercepted format.

5. The method of claim **1**, wherein intercepting packet data comprises:

intercepting the packet data based on a priority of the packet data.

6. The method of claim **1**, wherein the criteria for intercepting packet data includes whether packet data is traveling to or from an agent of a switch.

7. A network device, comprising:

a virtual insertion module to virtually insert a network appliance into any one of a plurality of logical points within a data path of a network, including a data tap descriptor module to define a data tap that describes one of the plurality of logical points in the data path for intercepting data packets in a network, and an application path descriptor module to define a first application path between the data tap and the network appliance;

an interception module to intercept data packets at the described logical point in the data path of the network based on an interception criteria;

forwarding circuitry to forward the intercepted data packets to the network appliance; and

a processor to implement the data tap descriptor and the application path descriptor module of the virtual insertion module, and the interception module.

8. The network device of claim **7**, wherein the data tap is dynamically updated with an updated interception criteria.

9. The network device of claim **7**, wherein the data tap descriptor module is further to:

define an ingress data tap to intercept a pre-routed form of a packet and an egress data tap to intercept a post-routed form of a packet.

10. The network device of claim **9**, wherein the data tap descriptor module is further to prevent an intercepted packet or a portion of an intercepted packet from being copied to a location other than the network appliance.

11. The network device of claim **7**, further comprising:

a re-interception prevention module to prevent a previously intercepted data packet from being re-intercepted by the data tap.

12. The network device of claim **7**, further comprising an injection module to return the intercepted data packets from the network appliance to the data path at the data tap.

13. A non-transitory computer-readable storage medium containing instructions that, when executed, cause a computer to:

virtually insert a network appliance in any one of a plurality of logical points within a data path of a network, including:

define a data tap that describes one of the plurality of logical points in the data path for intercepting data packets in the network;

define a first application path between the network appliance and the data tap; and

bind the first application path to the data tap.

14. The non-transitory computer-readable storage medium of claim **13**, comprising further instructions that cause the computer to:

intercept data packets at the described logical point in the data path based on an interception criteria; and

route the intercepted data packets to the network appliance.

15. The non-transitory computer-readable storage medium of claim **14**, comprising further instructions that cause the computer to:

return the intercepted data packets from the network appliance to the data path of the network in a pre-interception format.

16. The non-transitory computer-readable storage medium of claim **13**, comprising further instructions that cause the computer to:

prevent the intercepted data packets or a portion of the intercepted data packets from being copied to a location other than the network appliance.

17. The non-transitory computer-readable storage medium of claim **13**, wherein the instructions that cause the binding are implemented, at least in part, via one or more of a management information base (MIB) and a policy-based forwarding command line interface (CLI) syntax.

18. The non-transitory computer-readable storage medium of claim **13**, comprising further instructions to cause the computer to:

modify the data tap to describe a different logical point in the data path for intercepting data packets;

modify the first application path to define a new application path between the network appliance and the modified data tap.

19. The non-transitory computer-readable storage medium of claim **13**, wherein the instructions to define the application path comprise one or more of a direct memory access (DMA) instruction, a queuing instruction, a destination port, a destination virtual machine (VM), a packet encapsulation type.

20. The non-transitory computer-readable storage medium of claim **13**, further comprising instructions to cause the computer to:

define a second application path between a second application running on the network appliance and the data tap; and

bind the first application path to the second application path.

* * * * *